Terraform **provisioners** are a way to execute scripts or copy files on a resource after it’s created. They’re often used in **bootstrapping** (installing packages, configuring software), but HashiCorp recommends using them only when there’s no other way (better to use user\_data, Ansible, etc.).

Here’s the difference between **file**, **local-exec**, and **remote-exec** provisioners:

## 1. ****File Provisioner****

**Purpose**: Copies files/directories from your local machine (where you run Terraform) to the remote resource (like an EC2 instance).

**Execution Location**: Runs on the Terraform host → transfers files → remote instance.

**Use Case**: Copy configuration files, scripts, or binaries into the VM after provisioning.

## 2. ****Local-exec Provisioner****

**Purpose**: Runs a command **locally** on the machine where Terraform is executed.

**Execution Location**: Your **local system**, not on the remote instance.

**Use Case**: Notify services, update DNS, run Ansible, trigger shell scripts, etc.

## 3. ****Remote-exec Provisioner****

**Purpose**: Runs commands or scripts directly **on the remote resource** after it’s created.

**Execution Location**: Inside the remote machine (via SSH or WinRM).

**Use Case**: Install packages, configure software, run shell commands inside the instance.

### Key Differences

| **Provisioner** | **Runs Where?** | **Common Usage** |
| --- | --- | --- |
| **file** | Copies from **local → remote** | Upload configs, scripts |
| **local-exec** | On **Terraform host** | Trigger local scripts, update DNS, notify APIs |
| **remote-exec** | On **remote machine** | Run commands, install packages, bootstrap server |

**file** + **remote-exec** are often used together (upload a script → execute it).

* **local-exec** is useful when Terraform needs to interact with your **local environment** after a resource is created.
* When you use local-exec, Terraform is telling your **own computer** (the one where you run terraform apply) to run some command **after** the resource is created.
* It doesn’t run inside AWS or the server — it runs on **your laptop/PC**.

### Simple Example

* Imagine Terraform creates an **EC2 instance**.
* With **local-exec**, you could run a command on your computer that:
* Saves the new EC2 IP to a text file
* Sends a Slack/Teams notification
* Calls another script or tool (like Ansible, Docker, etc.)
* Runs **on your computer** (where you run terraform apply)
* Example: Save the EC2 server’s IP in a file on **your PC**.

local-exec = do something on my computer after Terraform makes a resource.

**file** + **remote-exec** are often used together (upload a script → execute it).

### ****remote-exec****

Runs **inside the server** (the remote machine that Terraform just created).

Example: After the EC2 server is ready, Terraform connects via SSH and runs commands like:

sudo apt update

sudo apt install nginx -y

| **Provisioner** | **Runs Where?** | **Simple Example** |
| --- | --- | --- |
| **local-exec** | On **your PC** | Save EC2 IP to a file on your computer |
| **remote-exec** | On **the EC2 instance** | Install software on the server |

· **local-exec = do something on my computer** after Terraform creates the server.

· **remote-exec = do something inside the server** after Terraform creates it.